A Knowledge Compilation Map for Conditional Preference Statements-based Languages

Hélène Fargier
IRIT-CNRS, Université de Toulouse
Toulouse, France
helene.fargier@irit.fr

Jérôme Mengin
IRIT-CNRS, Université de Toulouse
Toulouse, France
jerome.mengin@irit.fr

ABSTRACT

Conditional preference statements have been used to compactly represent preferences over combinatorial domains. They are at the core of CP-nets and their generalizations, and lexicographic preference trees. Several works have addressed the complexity of some queries (optimization, dominance in particular). We extend in this paper some of these results, and study other queries which have not been addressed so far, like equivalence, thereby contributing to a knowledge compilation map for languages based on conditional preference statements. We also introduce a new parameterised family of languages, which enables to balance expressiveness against the complexity of some queries.

KEYWORDS

Preferences; Knowledge Compilation; CP-nets

ACM Reference Format:
Hélène Fargier and Jérôme Mengin. 2021. A Knowledge Compilation Map for Conditional Preference Statements-based Languages. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Preference handling is a key component in several areas of Artificial Intelligence, notably for decision-aid systems. Research in Artificial Intelligence has led to the development of several languages that enable compact representation of preferences over complex, combinatorial domains. Some preference models rank alternatives according to their values given by some multivariate function; this is the case for instance with valued constraints [30], additive utilities and their generalizations [9, 25]. Ordinal models like CP nets and their generalisations [4, 8, 33], or lexicographic preferences and their generalisations [3, 10, 18, 22, 31, 34] use sets of conditional preference statements to represent a pre-order over the set of alternatives.

Many problems of interest, like comparing alternatives or finding optimal alternatives, are NP-hard for many of these models, even PSPACE hard for some models, which makes these representations difficult to use in some decision-aid systems like configurators, where real-time interaction with a decision maker is needed. One approach to tackle this problem is Knowledge Compilation, whereby a model, or a part of it, is *compiled*, off-line, into another representation which enables fast query answering, even if the compiled representation has a much bigger size. This approach has first been studied in propositional logic: [13, 14] compare how various subsets of propositional logic can succinctly, or not, express some propositional knowledge bases, and the complexity of queries of interest. [12] follow a similar approach to compare extensions of propositional logic which associate real values to models of a knowledge base; [19] provide such a map for value function-based models.

The aim of this paper is to initiate such a compilation map for models of preferences based on the language of conditional preference statements. We compare the expressiveness and succinctness of various languages on these conditional preference statements, and the complexity of several queries of interest for these languages.

The next section recalls some basic definitions about combinatorial domains and pre-orders, and introduces notations that will be used throughout. Section 3 gives an overview of various languages based on conditional preference statements that have been studied in the literature. We also introduce a new parameterised family of languages, which enables to balance expressiveness against the complexity of some queries. Section 4 and 5 respectively study expressiveness and succinctness for languages based on conditional preference statements. Sections 6 study the complexity of queries for these languages. Proofs can be found in [20].

2 PRELIMINARIES

2.1 Combinatorial Domain

We consider languages that can be used to represent the preferences of a decision maker over a combinatorial space $\underline{\mathcal{X}}$: here $\mathcal{X}$ is a set of attributes that characterise the possible alternatives, each attribute $X \in \mathcal{X}$ having a finite set of possible values $\underline{X}$; we assume that $|\underline{X}| \geq 2$ for every $X \in \mathcal{X}$; then $\underline{\mathcal{X}}$ denotes the cartesian product of the domains of the attributes in $\mathcal{X}$, its elements are called alternatives. For a binary attribute X, we will often denote by $x, \bar{x}$ its two possible values. In the sequel, n is the number of attributes in $\mathcal{X}$.

For a subset U of $\mathcal{X}$, we will denote by $\underline{U}$ the cartesian product of the domains of the attributes in U, called instantiations of U, or partial instantiations (of $\mathcal{X}$). If v is an instantiation of some $V \subseteq \mathcal{X}$, $v[U]$ denotes the restriction of v to the attributes in $V \cap U$; we say that instantiation $u \in \underline{U}$ and v are compatible if $v[U \cap V] = u[U \cap V]$; if $U \subseteq V$ and $v[U] = u$, we say that v extends u.

Sets of partial instantiations can often be conveniently, and compactly, specified with propositional formulas: the atoms are $X = x$ for every $X \in \mathcal{X}$ and $x \in \underline{X}$, and we use the standard connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence) and $\neg$ (negation); we denote by $\top$ (resp. $\bot$) the formula always true (resp. false). Implicitly, this propositional logic is equipped with a theory that enforces that every attribute has precisely one value

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

from its domain; so, for two distinct values x, x' of attribute X, the formula $X = x \wedge X = x'$ is a contradiction; also, the interpretations are thus in one-to-one correspondence with $\underline{\mathcal{X}}$. If α is such a propositional formula over $\mathcal{X}$ and $o \in \underline{\mathcal{X}}$, we will write $o \models \alpha$ when o satisfies α, that is when, assigning to every literal $X = x$ that appears in α the value true if $o[X] = x$, and the value false otherwise, makes α true.

Given a formula α, or a partial instantiation u, $\mathrm{Var}(\alpha)$ and $\mathrm{Var}(u)$ denote the set of attributes, the values of which appear in α and u respectively.

When it is not ambiguous, we will use x as a shorthand for the literal $X = x$; also, for a conjunction of such literals, we will omit the $\wedge$ symbol, thus $X = x \wedge Y = \bar{y}$ for instance will be denoted $x\bar{y}$.

2.2 Preference Relations

Depending on the knowledge that we have about a decision maker's preferences, given any pair of distinct alternatives $o, o' \in \underline{\mathcal{X}}$, one of the following situations must hold: one may be strictly preferred over the other, or o and o' may be equally preferred, or o and o' may be incomparable.

Assuming that preferences are transitive, such a state of knowledge about the DM's preferences can be characterised by a preorder $\succeq$ over $\underline{\mathcal{X}}$: $\succeq$ is a binary, reflexive and transitive relation; for alternatives o, o', we then write $o \succeq o'$ when $(o, o') \in \succeq$; $o \succ o'$ when $(o, o') \in \succeq$ and $(o', o) \notin \succeq$; $o \sim o'$ when $(o, o') \in \succeq$ and $(o', o) \in \succeq$; $o \bowtie o'$ when $(o, o') \notin \succeq$ and $(o', o) \notin \succeq$. Note that for any pair of alternatives $o, o' \in \underline{\mathcal{X}}$ either $o \succ o'$, or $o' \succ o$, or $o \sim o'$ or $o \bowtie o'$.

The relation $\sim$ defined in this way is the *symmetric part* of $\succeq$, it is reflexive and transitive, $\bowtie$ is irreflexive, they are both symmetric. The relation $\succ$ is the *irreflexive part* of $\succeq$, it is what is usually called a strict partial order: it is irreflexive and transitive.

Terminology and notations. We say that alternative o *dominates* alternative o' (w.r.t. $\succeq$) if and only if $o \succeq o'$; if $o \succ o'$, then we say that o *strictly dominates* o'. We use standard notations for the complements of $\succ$ and $\succeq$: we write $o \not\succeq o'$ when it is not the case that $o \succeq o'$, and $o \not\succ o'$ when it is not the case that $o \succ o'$.

3 LANGUAGES

3.1 Conditional Preference Statements

A *conditional preference statement* (aka., CP statement) over $\mathcal{X}$ is an expression of the form $\alpha \mid V : w \geq w'$, where α is a propositional formula over $U \subseteq \mathcal{X}$, $w, w' \in \underline{W}$ are such that $w[X] \neq w'[X]$ for every $X \in W$, and U, V, W are disjoint subsets of $\mathcal{X}$, not necessarily forming a partition of $\mathcal{X}$. Informally, such a statement represents the piece of knowledge that, when comparing alternatives o, o' that both satisfy α, the one that has values w for W is preferred to the one that has values w' for W, irrespective of the values of the attributes in V, every attribute in $\mathcal{X} \setminus (V \cup W)$ being fixed. We call α the conditioning part of the statement; we call W the swapped attributes, and V the free part.

Example 1 ((Example A in [35], slightly extended)). Consider planning a holiday, with three choices / attributes: wait til next month ($W = w$) or leave now ($W = \bar{w}$), going to city 1, 2 or 3 ($C = c_1, C = c_2$ or $C = c_3$), travelling by plane ($P = p$) or by car ($P = \bar{p}$). I would rather go now, irrespective of the other attributes: $\top \mid \{CP\} : \bar{w} \geq w$. All else being equal, I prefer to go to city 3, city 1 being my second best choice: $\top \mid \emptyset : c_3 \geq c_1 \geq c_2$. Also, if I go now, I prefer to fly: $\bar{w} \mid \emptyset : p \geq \bar{p}$. Together, the last two statements imply that if I go now, I prefer to go to city 3 by plane than go to city 1 by car; however these statements do not say what I prefer between flying to city 1 or driving to city 3. In fact, I prefer the former, this *tradeoff* can be expressed with the statement $\bar{w} \mid \emptyset : c_1 p \geq c_3 \bar{p}$. Finally, if I go later, I prefer to drive, irrespective of the city: $w \mid \{C\} : \bar{p} \geq p$.

Conditional preference statements have been studied in many works, under various language restrictions. They are the basis for CP-nets [4, 6] and their extensions, and have been studied in a more logic-based fashion by e.g. [24] and [32, 33, 35]. [1] They are closely related to *CI-statements* by [7]

For the semantics sets of CP statements, we use the definitions of [35]. Given a statement $\alpha \mid V : w \geq w'$, let $U = \mathrm{Var}(\alpha)$ and $W = \mathrm{Var}(w) = \mathrm{Var}(w')$: a *worsening swap* is any pair of alternatives (o, o') such that $o[U] = o'[U] \models \alpha$, $o[W] = w$ and $o'[W] = w'$, and such that for every attribute $Y \notin U \cup V \cup W$ it holds that $o[Y] = o'[Y]$; we say that $\alpha \mid V : w \geq w'$ *sanctions* (o, o'). For a set of CP-statements φ, let φ^* be the set of all worsening swaps sanctioned by statements of φ, and define $\succeq_\varphi$ to be the reflexive and transitive closure of φ^*. [35] proves that $o \succeq_\varphi o'$ if and only if $o = o'$ or φ^* contains a finite sequence of worsening swaps $(o_i, o_{i+1})_{0 \leq i \leq k-1}$ with $o_0 = o$ and $o_k = o'$.[2]

Example 2 (Example 1, continued). Let $\varphi = \{\top \mid \{CP\} : \bar{w} \geq w, \top \mid \emptyset : c_3 \geq c_1 \geq c_2, n \mid \emptyset : p \geq \bar{p}, \bar{w} \mid \emptyset : c_1 p \geq c_3 \bar{p}, w \mid \{C\} : \bar{p} \geq p\}$. Then $\top \mid \{CP\} : \bar{w} \geq w$ sanctions for instance $(\bar{w}c_2p, wc_3\bar{p})$, so $\bar{w}c_2p \succeq_\varphi wc_3\bar{p}$. Also, $\top \mid \emptyset : c_3 \geq c_1 \geq c_2$ sanctions $(\bar{w}c_1p, \bar{w}c_2p)$, $\bar{w} \mid \emptyset : p \geq \bar{p}$ sanctions $(\bar{w}c_2p, \bar{w}c_2\bar{p})$, so, by transitivity, $\bar{w}c_1p \succeq_\varphi \bar{w}c_2\bar{p}$. It is not difficult to check that $\bar{w}c_2p \bowtie_\varphi \bar{w}c_1\bar{p}$.

Let us call CP the language where formulas are sets of statements of the general form $\alpha \mid V : w \geq w'$. This language is very expressive: it is possible to represent any preorder "in extension" with preference statements of the form $o \geq o'$ – they all have $W = \mathcal{X}$ as set of swapped attributes, $\alpha = \top$ as condition, and no free attribute.

This expressiveness has a cost: we will see that many queries about pre-orders represented by CP-statements are PSPACE-hard for the language CP. Several restrictions / sub-languages have been studied in the literature, we review them below.

Linearisability. Although the original definition of CP-nets by [6] does not impose it, many works on CP-nets, especially following [4], consider that they are intended to represent a strict partial order, that is, that $\succeq_\varphi$ should be antisymmetric; equivalently, this means that the irreflexive part $\succ_\varphi$ of $\succeq_\varphi$ can be extended to a linear order. We say that a set φ of CP-statements is *linearisable* in this case.[3]

[1] The formula $u \mid V : x \geq x'$ is written $u : x > x'[V]$ by [35].

[2] Actually, [35] proves that (o, o') is in the transitive closure of φ^* if and only there is such a worsening sequence from o to o', but adding the reflexive closure to this transitive closure does not change the result, since we can add any pair (o, o) to, or remove it from, any sequence of worsening swaps without changing the validity of the sequence.

[3] Such sets of CP-statements are often called *consistent* in the standard terminology on CP-nets, but we prefer to depart from this definition which only makes sense when one asserts that φ should indeed represent a strict partial order.

Notations. We write $\alpha : w \geq w'$ when V is empty, and $w \geq w'$ when V is empty and $\alpha = \top$. Note that we reserve the symbol $\geq$ for conditional preference statements, whereas "curly" symbols $\succ$, $\not\succ$, $\succeq$, $\not\succeq$ are used to represent relations over the set of alternatives.

In the remainder of this section, we present various sublanguages of CP. Some are defined by imposing various simple syntactical restrictions on the formulas, two are languages which have been well studied (CP-nets and lexicographic preference trees); we close the section by introducing a new, parameterised class of sublanguages of CP which have interesting properties, as will be shown in subsequent subsections.

3.2 Statement-wise Restrictions

Some restrictions are on the syntactical form of statements allowed; they bear on the size of the set of free attributes, or on the size of the set of swapped attributes, or on the type of conditioning formulas allowed. Given some language $\mathcal{L} \subseteq$ CP, we define the following restrictions:

$\mathcal{L}\not\rhd$ = only formulas with empty free parts ($V = \emptyset$) for every statement;[4]

$\mathcal{L}\wedge$ = only formulas where the condition α of every statement is a conjunction of literals;

k-$\mathcal{L}$ = only formulas where the set of swapped attributes contains no more than k attributes ($|\, W \,| \leq k$) for every statement; in particular, we call elements of 1-CP *unary* statements.

In particular, 1-CP$\wedge$ is the language studied by [35], and 1-CP$\not\rhd$ is the language of generalized CP-nets as defined by [24].

3.3 Graphical Restrictions

Given $\varphi \in$ CP over set of attributes $\mathcal{X}$, we define D_φ as the graph with sets of vertices $\mathcal{X}$, and such that there is an edge (X, Y) if there is $\alpha \,|\, V : w \geq w' \in \varphi$ such that $X \in \mathrm{Var}(\alpha)$ and $Y \in \mathrm{Var}(w)$, or $X \in \mathrm{Var}(w)$ and $Y \in V$. We call D_φ the *dependency graph* of φ. Note that D_φ can be computed in polynomial time. This definition, inspired by [35, Def. 15], generalises that of [4], which is restricted to the case where all CP statements are unary and have no free attributes, and that of [8], who study statements with free attributes. Many tractability results on sets of CP statements have been obtained when D_φ has good properties. Given some language $\mathcal{L} \subseteq$ CP, we define:

$\mathcal{L}\not\circlearrowleft$ = the restriction of $\mathcal{L}$ to *acyclic* formulas, which are those φ such that D_φ is acyclic;[5]

$\mathcal{L}\not\circlearrowleft$**poly** = the restriction of $\mathcal{L}$ to formulas where the dependency graph is a polytree.

[35] also defines a weaker graphical restriction, called "context-uniform conditional acyclicity", but it turns out that it does gives rise to the same complexities as another, weaker restriction called "conditional acyclicity" by [35], which we generalize in section 3.6.

[4] In the literature, the symbol $\rhd$ is sometimes used to represent an *importance* relation between attributes; and, as explained by [35], statement $\alpha \,|\, V : w \geq w'$ is a way to express that attributes in $\mathrm{Var}(w)$ are more important than those in V (when α is true).

[5] This is *full acyclicity* in [35].

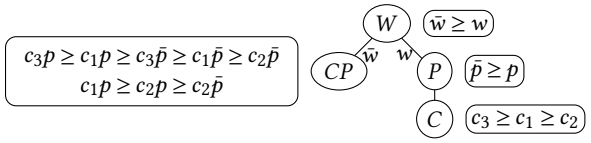


Figure 1: An LP-tree equivalent to the set of CP-statements of Example 2.

3.4 CP-nets

In their seminal work, [4] define a CP-net over a set of attributes $\mathcal{X}$ to be composed of two elements:

(1) a directed graph over $\mathcal{X}$, which should represent *preferential dependencies* between attributes;[6]
(2) a set of conditional preference tables, one for every attribute X: if U is the set of parents of X in the graph, the conditional preference table for X contains exactly $|\, \underline{U} \,|$ rules $u : \geq$, for every $u \in \underline{U}$, where the $\geq$'s are linear orders over $\underline{X}$.

Therefore, as shown by [35], CP-nets can be seen as sets of unary CP statements in conjunctive form with no free attribute. Specifically, given a CP-net $\mathcal{N}$ over $\mathcal{X}$, define $\varphi_\mathcal{N}$ to be the set of all CP statements $u : x \geq x'$, for every attribute X, every $u \in \underline{U}$ where U is the set of parents of X in the graph, every $x, x' \in \underline{X}$ such that x, x' are consecutive values in the linear order $\geq$ specified by the rule $u : \geq$ of $\mathcal{N}$. Then the dependency graph of $\varphi_\mathcal{N}$, as defined in Section 3.3, coincides with the graph of $\mathcal{N}$. We call

CPnet = the language that contains all $\varphi_\mathcal{N}$, for every CP-net $\mathcal{N}$.

Note that CPnet $\subseteq$ 1-CP$\wedge\not\rhd$. For a given $\varphi \in$ 1-CP$\wedge\not\rhd$, being a CP-net necessitates a very strong form of local consistency and completeness: for every attribute X with parents U in D_φ, for every $u \in \underline{U}$, for every $x, x' \in \underline{X}$, φ must explicitly, and uniquely, order ux and ux'.

[8] define TCP-nets as an extension of CP-nets where it is possible to represent tradeoffs, by stating that, under some conditions, some attributes are more important than other ones. [35] describes how TCP-nets can be transformed, in polynomial time, into equivalent sets of 1-CP$\wedge$ statements.

3.5 Lexicographic Preference Trees

LP-trees generalise lexicographic orders, which have been widely studied in decision making – see e.g. [21]. As an inference mechanism, they are equivalent to search trees used by [5], and formalised by [32, 35]. As a preference representation, and elicitation, language, slightly different definitions for LP-trees have been proposed by [3, 10, 18]. We use here a definition which subsumes the others.

An LP-tree that is equivalent to the set of CP-statements of Example 2 is depicted on Figure 1. More generally, an LP-tree over $\mathcal{X}$ is a rooted tree with labelled nodes and edges, and a set of preference tables; specifically

- every node N is labelled with a set of attributes, denoted $\mathrm{Var}(N)$;
- if N is not a leaf, it can have one child, or $|\, \underline{\mathrm{Var}(N)} \,|$ children;

[6] Given some pre-order $\geq$ over $\mathcal{X}$, attribute X is said to be preferentially dependent on attribute Y if there exist $x, x' \in \underline{X}$, $y, y' \in \underline{Y}$, $z \in \underline{\mathcal{X} \setminus (\{X, Y\})}$ such that $xyz \geq_\varphi x'yz$ but $xy'z \not\geq_\varphi x'y'z$.

- in the latter case, the edges that connect N to its children are labelled with the instantiations in $\underline{\mathsf{Var}(N)}$;
- if N has one child only, the edge that connects N to its child is not labelled: all instantiations in $\underline{\mathsf{Var}(N)}$ lead to the same subtree;
- we denote by $\mathsf{Anc}(N)$ the set of attributes that appear in the nodes between the root and N (excluding those at N), and by $\mathsf{Inst}(N)$ (resp. $\mathsf{NonInst}(N)$) the set of attributes that appear in the nodes above N that have more than one children (resp. only one child);
- a conditional preference table $\mathsf{CPT}(N)$ is associated with N: it contains local preference rules of the form $\alpha : \geq$, where $\geq$ is a preorder over $\underline{\mathsf{Var}(N)}$, and α is a propositional formula over some attributes in $\mathsf{NonInst}(N)$.

We assume that the rules in $\mathsf{CPT}(N)$ define their preorder over $\underline{\mathsf{Var}(N)}$ in extension. Additionally, two constraints guarantee that an LP-tree φ defines a unique preorder over $\underline{\mathcal{X}}$:

- no attribute can appear at more than one node on any branch of φ; and,
- at every node N of φ, for every $u \in \underline{\mathsf{NonInst}(N)}$, $\mathsf{CPT}(N)$ must contain exactly one rule $\alpha : \geq$ such that $u \models \alpha$.

Given an LP-tree φ and an alternative $o \in \underline{\mathcal{X}}$, there is a unique way to traverse the tree, starting at the root, and along edges that are either not labelled, or labelled with instantiations that agree with o, until a leaf is reached. Now, given two distinct alternatives o, o', it is possible to traverse the tree along the same edges as long as o and o' agree, until a node N is reached which is labelled with some W such that $o[W] \neq o'[W]$: we say that N decides $\{o, o'\}$.

In order to define $\succeq_\varphi$ for some LP-tree φ, let φ^* be the set of all pairs of distinct alternatives (o, o') such that there is a node N that decides $\{o, o'\}$ and the only rule $\alpha : \geq \in \mathsf{CPT}(N)$ with $o[\mathsf{NonInst}(N)] = o'[\mathsf{NonInst}(N)] \models \alpha$ is such that $o[W] \geq o'[W]$. Then $\succeq_\varphi$ is the reflexive closure of φ^*.

Proposition 1. *Let φ be an LP-tree over $\mathcal{X}$, then $\succeq_\varphi$ as defined above is a preorder. Furthermore, $\succeq_\varphi$ is a linear order if and only if 1) every attribute appears on every branch and 2) every preference rule specifies a linear order.*

An LP-tree φ is said to be *complete* if the two conditions in Proposition 1 hold, that is, if $\succeq_\varphi$ is a linear order.

From a semantic point of view, an LP-tree φ is equivalent to the set that contains, for every node N of φ labelled with $W = \mathsf{Var}(N)$, and every rule $\alpha : \geq^\alpha_N$ in $\mathsf{CPT}(N)$, all CP statements of the form $\alpha \wedge u \mid V : w^\sharp \geq w'^\sharp$, where

- u is the combination of values given to the attributes in $\mathsf{Inst}(N)$ along the edges between the root and N, and
- $w, w' \in \underline{W}$ such that $w \geq^\alpha_N w'$, and $W^\sharp$ is the set of attributes on which w and w' have distinct values, and $w^\sharp = w[W^\sharp]$, and $w'^\sharp = w'[W^\sharp]$; and
- $V = [\mathcal{X} - (\mathsf{Anc}(N) \cup W)]$.

This set of statements indicate that outcomes that agree on $\mathsf{Anc}(N)$ and satisfy $u \wedge \alpha$, but have different values for $\mathsf{Var}(N)$, should be ordered according to $\geq^\alpha_N$, whatever their values for attributes in V.

LPT = the language of LP-trees as defined above; we consider that LPT is a subset of CP.[7]

Note that, using the notations defined above, k-LPT = LPT$\cap k$-CP is the restriction of LPT where every node has at most k attributes, for every $k \in \mathbb{N}$; in particular, 1-LPT is the language of LP-trees with one attribute at each node; and LPT$\wedge$ = LPT $\cap$ CP$\wedge$ is the restriction of LPT where the condition α in every rule at every node is a conjunction of literals. Search trees of [32, 35] and LP-trees as defined by [3, 27] are sublanguages of 1-LPT$\wedge$; LP-trees of [18] and [10] are sublanguages of LPT$\wedge$.

3.6 Lexico-compatible Formulas

Many graphical restrictions that have been proposed in order to enable polytime answers to some queries are in fact particular cases of a more general property which we study now. We define a new, parameterised family of languages. Given some language $\mathcal{L} \subseteq$ CP and $k \in \mathbb{N}$, we define:

$\mathcal{L}\mathbb{O}^{\mathbf{lex}}_k$ = the restriction of $\mathcal{L}$ to formulas φ such that there exists some complete LP-tree $\psi \in k$-LPT such that $\succeq_\psi$ extends $\succeq_\varphi$.

We say that formulas of $\mathsf{CP}\mathbb{O}^{\mathsf{lex}}_k$ are *k-lexico-compatible*.[8]

[35] proves that acyclic formulas of 1-CP are 1-lexico-compatible when they enjoy some local consistency property; it illustrates that k-lexico-compatibility is indeed a weak form of acyclicity. We will see that k-lexico-compatibility makes some queries tractable.

The next result shows that proving that some $\varphi \in$ CP is k-lexico-compatible, for a fixed k, is not always easy:

Proposition 2. *For a fixed $k \in \mathbb{N}$, checking if a formula $\varphi \in$ CP is k-lexico-compatible is coNP-complete.*

Algorithm 1 checks if a given formula is k-lexico-compatible. Given $\varphi \in$ CP, it builds, in a top-down fashion, a complete $\psi \in$ k-LPT that is compatible with φ. The algorithm is similar to the algorithm proposed by [3] to learn an LP-tree that sanctions a given set of pairs (o, o'). It starts with an empty root node at step 1; then, while there is some empty node, it picks one of them, call it N, and calls at step 2b the function chooseAttribute to get a pair $(T, \geq)$ to label N, where T is a set of at most k attributes, none of which appear above N, and $\geq$ is a linear order over $\underline{T}$; if no such pair is compatible with φ, in a sense that will be defined shortly, chooseAttribute returns failure and the algorithm stops at step 2c; otherwise, if there remain some attributes that do not appear in T nor at any node above N, then the algorithm expands the tree below N at step 2e by creating a branch and a new node for every instantiation $t \in \underline{T}$, and loops.

Note that all edges of the tree built by the algorithm are labelled, so that, at every node N, $\mathsf{NonInst}(N) = \emptyset$, so $\mathsf{CPT}(N)$ must contain only one rule of the form $\top : \geq$, where $\top$ is the formula always true. This is why chooseAttribute needs to return one linear order over $\underline{T}$ only, we do not need to specify the trivial condition $\top$ here. There may be a more compact k-LP-tree compatible with φ than the one returned by the above algorithm when it does not fail, but we are only interested here in checking if φ is k-lexico-compatible, and we

[7] Strictly speaking, for LPT $\subseteq$ CP to hold, we can add the possibility to augment every formula in CP with a tree structure.

[8] This definition generalises *conditionally acyclic* formulas of [35], which are the formulas of $\mathsf{CP}\mathbb{O}^{\mathsf{lex}}_1$.

Algorithm 1: Build complete LP tree

Input: $\varphi \in$ CP; $k \in \mathbb{N}$;
Output: $\psi \in$ k-LPT, ψ complete, s.t. $\succeq_\psi \supseteq \succeq_\varphi$, or FAILURE;
(1) $\psi \leftarrow$ {an unlabelled root node};
(2) while ψ contains some unlabelled node:
  (a) choose unlabelled node N of ψ;
  (b) $(T, \geq) \leftarrow$ chooseAttribute(N, k, φ);
  (c) if $T =$ FAILURE then STOP and return FAILURE;
  (d) label N with $(T, \geq)$;
  (e) if $\mathrm{Anc}(N) \cup T \neq \mathcal{X}$, for each $t \in \underline{T}$: add new unlabelled node to ψ, attached to N with edge labelled with t;
(3) return ψ.

have seen that the problem is coNP-complete, so it seems difficult to avoid exploring a tree with size exponential in the size of φ in the worst case. We now specifiy some condition that chooseAttribute must verify in order for the algorithm to be correct and complete. Given any yet unlabelled node N of the tree being build, let $\varphi(N) = \{\alpha \mid V : w \geq w' \in \varphi \mid \alpha \wedge \mathrm{inst}(N) \not\models \bot, W \cap \mathrm{Anc}(N) = \emptyset\}$.

Definition 1. We say that chooseAttribute is φ-compatible if the pair $(T, \geq)$ that chooseAttribute returns at some yet unlabelled node N is such that for every $\alpha \mid V : w \geq w' \in \varphi(N)$: (1) if $\mathrm{Var}(w) \cap T = \emptyset$, then $V \cap T = \emptyset$; (2) if $\mathrm{Var}(w) \cap T \neq \emptyset$, then $t >^N t'$ for every $t, t' \in \underline{T}$ such that $t \wedge w \not\models \bot$, $t' \wedge w' \not\models \bot$, $t[\mathcal{X} \setminus (V \cup W)] = t'[\mathcal{X} \setminus (V \cup W)]$ and $t \wedge \alpha \not\models \bot$. If no such pair $(T, \geq)$ can be found, then chooseAttribute must return failure.

Condition (2) guarantees that N will correctly decide every pair of alternatives that is sanctionned by $\alpha \mid V : w \geq w'$ and that will be decided at N. When the entire tree is built in this way, condition (1) guarantees that at every node N, if $\alpha \mid V : w \geq w' \in \varphi(N)$ then $V \cap \mathrm{Anc}(N) = \emptyset$.

Proposition 3. *Given $\varphi \in$ CP and some $k \in \mathbb{N}$, suppose that chooseAttribute is φ-compatible, then $\varphi \in CP\not\diamond_k^{lex}$ if and only if the algorithm above returns some $\psi \in$ k-LPT such that $\succeq_\psi \supseteq \succeq_\varphi$; otherwise, it returns FAILURE.*

Note that chooseAttribute can be implemented to run in polynomial time, for fixed k: there are no more than $\sum_{i=1}^{k} \binom{n}{i} \leq kn^k$ possibilities for the T it can return, and the number of pairs t, t' that it must check against every statement in $\varphi(N)$ is bounded by $|\underline{T}|^2$, and $|T|$ is bounded by d^k, where d is the size of the largest domain of the attributes in $\mathcal{X}$. Also, each branch of the tree returned by the algorithm, when it succeeds, can have at most n nodes, but the tree can have up to d^n leaves.

4 EXPRESSIVENESS

We detail our results about expressiveness of the various languages studied here in this section, the results about succinctness are in the next section. These results are summarised on Figure 2.

Definition 2. Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that $\mathcal{L}$ is *at least as expressive as* $\mathcal{L}'$, written $\mathcal{L} \sqsupseteq \mathcal{L}'$, if every preorder that can be represented with a formula of $\mathcal{L}'$ can also be represented with a formula of $\mathcal{L}$; we write $\mathcal{L} \sqsupset \mathcal{L}'$ if $\mathcal{L} \sqsupseteq \mathcal{L}'$ but it is not the case that $\mathcal{L}' \sqsupseteq \mathcal{L}$, and say in this case that $\mathcal{L}$ is strictly more expressive than $\mathcal{L}'$. We write $\mathcal{L} \sqsupset\sqsubset \mathcal{L}'$ when the two languages are equally expressive.

We reserve the usual "rounded" symbols $\subset$ and $\subseteq$ for (strict) set inclusion, and $\supset$ and $\supseteq$ for the reverse inclusions. Note that $\sqsupseteq$ is a preorder, and obviously $\mathcal{L} \supseteq \mathcal{L}'$ implies $\mathcal{L} \sqsupseteq \mathcal{L}'$.

Clearly, CP$\not\diamond$ $\subset$ CP and CP$\wedge$ $\subset$ CP; however, these three languages have the same expressiveness, because of the following:

Property 4. *Given some preorder $\geq$, define $\varphi \in \{o[\Delta(o, o')] \geq o'[\Delta(o, o')] \mid o \geq o', o \neq o'\}$, where $\Delta(o, o')$ is the set of attributes that have different values in o and o', then $\varphi \in CP\not\diamond \cap CP\wedge$, and $\succeq_\varphi = \geq$.*

A large body of works on CP-statements since the seminal paper by [5] concentrate on various subsets of 1-CP. With this strong restriction on the number of swapped attributes, CP-statements have a reduced expressiveness.

Example 3. Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\geq$ such that $ab \succ \bar{a}\bar{b} \succ a\bar{b} \succ \bar{a}b$. This can be represented in CP with $\varphi = \{ab \geq \bar{a}\bar{b}, \bar{a}\bar{b} \geq a\bar{b}, a\bar{b} \geq \bar{a}b\}$. But it cannot be represented in 1-CP: $\{b : a \geq \bar{a}, \bar{b} : \bar{a} \geq a, a : b \geq \bar{b}, \bar{a} : \bar{b} \geq b\}^* \subseteq \varphi^*$, but this is not sufficient to compare $a\bar{b}$ with $\bar{a}b$. The four remaining formulas of 1-CP over these two attributes are $B : a \geq \bar{a}$, $B : \bar{a} \geq a$, $A : b \geq \bar{b}$, $A : \bar{b} \geq b$, adding any of them to φ yields a preorder which would not be antisymmetric.

Forbidding free parts incurs an additional loss in expressiveness:

Example 4. Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\geq$ such that $ab \succ a\bar{b} \succ \bar{a}b \succ \bar{a}\bar{b}$. This can be represented in 1-CP with $\varphi = \{B : a \geq \bar{a}, b \geq \bar{b}\}$. But the "tradeoff" $a\bar{b} \succ \bar{a}b$ cannot be represented in 1-CP$\not\diamond$, any formula of 1-CP$\not\diamond$ that implies it will put some *intermediate* alternative between $a\bar{b}$ and $\bar{a}b$

However, restricting to conjunctive statements does not incur a loss in expressiveness.

Proposition 5. *$CP = \bigcup_{k \in \mathbb{N}}$ k-CP and, for every $k \in \mathbb{N}$:*

$$CP\wedge \sqsupset\sqsubset CP\not\diamond \sqsupset\sqsubset CP \sqsupset k\text{-}CP \sqsupset\sqsubset k\text{-}CP\wedge \sqsupset k\text{-}CP\not\diamond \sqsupset\sqsubset k\text{-}CP\wedge\not\diamond$$
$$k\text{-}CP \sqsupset (k\text{-}1)\text{-}CP.$$

Because an LP-tree can be a single node labelled with $\mathcal{X}$, and a single preference rule $\top : \geq$ where $\geq$ can be any preorder, LPT can represent any preorder. Limiting to conjunctive conditions in the rules is not restrictive. However, restricting to 1-LPT reduces expressiveness, even if one considers formulas of 1-CP that represent total, linear orders:

Example 5. Let $\varphi = \{a \geq \bar{a}, \bar{c} \mid A : \bar{b} \geq b, \bar{a}c : \bar{b} \geq b, ac : b \geq \bar{b}, a : c \geq \bar{c}, \bar{a} \mid B : \bar{c} \geq c\}$. This yields the following linear order: $abc \succeq_\varphi a\bar{b}c \succeq_\varphi a\bar{b}\bar{c} \succeq_\varphi \bar{a}\bar{b}\bar{c} \succeq_\varphi ab\bar{c} \succeq_\varphi \bar{a}b\bar{c} \succeq_\varphi \bar{a}\bar{b}c \succeq_\varphi \bar{a}bc$. No $\psi \in$ 1-LPT can represent it: A could not be at the root of such a tree because for instance $a\bar{b}\bar{c} \succeq_\varphi \bar{a}\bar{b}\bar{c}$ and $\bar{a}\bar{b}\bar{c} \succeq_\varphi ab\bar{c}$; neither could C, since $a\bar{b}c \succeq_\varphi a\bar{b}\bar{c}$ and $\bar{a}b\bar{c} \succeq_\varphi \bar{a}\bar{b}c$; and finally B could not be at the root either, because $abc \succeq_\varphi a\bar{b}c$ and $\bar{a}\bar{b}\bar{c} \succeq_\varphi ab\bar{c}$.

Proposition 6. *$LPT = \bigcup_{k \in \mathbb{N}}$ k-LPT and, for every $k \in \mathbb{N}$:*

$$CP \sqsupset\sqsubset LPT \sqsupset\sqsubset LPT\wedge \sqsupset k\text{-}LPT \sqsupset\sqsubset k\text{-}LPT\wedge \sqsupset (k\text{-}1)\text{-}LPT.$$

Finally, note that k-lexico-compatibility is a weaker restriction than being a k-LP-tree.

Proposition 7. *For every $k \in \mathbb{N}$: $CP\emptyset_k^{lex} \sqsupset CP\emptyset_{k-1}^{lex}$, and $CP\emptyset_k^{lex} \sqsupset$ k-LPT.*

[35] proves that 1-CP$\emptyset \subseteq$ CP$\emptyset_1^{\text{lex}}$. Whether this property can be generalised, with an appropriate definition of *k-acyclicity*, is left for future work.

5 SUCCINCTNESS

Another criterion is the relative sizes of formulas that can represent the same preorder in different languages. [11] study the space efficiency of various propositional knowledge representation formalisms. An often used definition of succinctness [14, 23] makes it a particular case of expressiveness, which is not a problem when comparing languages of same expressiveness. However, we study here languages with very different expressiveness, so we need a more fine grained definition:

Definition 3. Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that $\mathcal{L}$ is *at least as succinct as* $\mathcal{L}'$, written $\mathcal{L} \leqq \mathcal{L}'$, if there exists a polynomial p such that for every $\varphi' \in \mathcal{L}'$, there exists $\varphi \in \mathcal{L}$ that represent the same preorder as φ' and such that $|\varphi| < p(|\varphi'|)$.[9] Moreover, we say that $\mathcal{L}$ is *strictly more succinct than* $\mathcal{L}'$, written $\mathcal{L} \ll \mathcal{L}'$, if $\mathcal{L} \leqq \mathcal{L}'$ and for every polynomial p, there exists $\varphi \in \mathcal{L}$ such that:

- there exists $\varphi' \in \mathcal{L}'$ such that $\succeq_\varphi = \succeq_{\varphi'}$, but
- for every $\varphi' \in \mathcal{L}'$ such that $\succeq_\varphi = \succeq_{\varphi'}$, $|\varphi'| > p(|\varphi|)$.

With this definition, $\mathcal{L} \ll \mathcal{L}'$ if every formula of $\mathcal{L}'$ has an equivalent formula in $\mathcal{L}$ which is "no bigger" (up to some polynomial transformation of the size of φ), and there is at least one sequence of formulas (one formula for every polynomial p) in $\mathcal{L}$ that have equivalent formulas in $\mathcal{L}'$ but necessarily "exponentially bigger".[10]

Proposition 8. *The following hold, for languages $\mathcal{L}$, $\mathcal{L}'$, $\mathcal{L}''$:*

- *if $\mathcal{L} \supseteq \mathcal{L}'$ then $\mathcal{L} \leqq \mathcal{L}'$; and if $\mathcal{L} \leqq \mathcal{L}'$, then $\mathcal{L} \sqsupseteq \mathcal{L}'$;*
- *if $\mathcal{L} \ll \mathcal{L}'$ then $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$;*
- *if $\mathcal{L} \sqsubset \mathcal{L}'$, the reverse implication holds: if $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$ then $\mathcal{L} \ll \mathcal{L}'$ (otherwise, it might be that $\mathcal{L}' \not\leqq \mathcal{L}$ because $\mathcal{L}' \not\sqsupseteq \mathcal{L}$);*
- *if $\mathcal{L} \supseteq \mathcal{L}'$ and $\mathcal{L}' \ll \mathcal{L}''$, then $\mathcal{L} \ll \mathcal{L}''$.*

Restricting the conditioning part of the statements to be conjunctions of literals does induce a loss in succinctness.

Example 6. Consider $2n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_n, Z$, and let φ contain $2n+2$ unary CP-statements with no free attribute: $(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \ldots \wedge (x_n \vee y_n) : z \geq \bar{z}$, $\neg[(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \ldots \wedge (x_n \vee y_n)] : \bar{z} \geq z$ and $\bar{x}_i \geq x_i$ and $\bar{y}_i \geq y_i$ for every $i \in \{1, \ldots, n\}$. Then $\varphi \in$ 1-CP$\not\wedge$, but φ is not in conjunctive form. A set of conjunctive CP-statements equivalent to φ has to contain all 2^n statements of the form $\mu_1\mu_2 \ldots \mu_n : z \geq \bar{z}$ with $\mu_i = x_i$ or $\mu_i = y_i$ for every i.

[9] Where $|\varphi| = \sum_{\alpha \mid V : w \geq w' \in \varphi} (|\alpha| + |V| + 2 |\text{Var}(w)|)$, with $|\alpha|$ = the number of connectives plus the number of atoms of α.

[10] When $\ll$ is defined as the strict counterpart of $\leqq$, it can happen that $\mathcal{L} \ll \mathcal{L}'$ even if there is no real difference in representation size in the two languages, but $\mathcal{L} \sqsupset \mathcal{L}'$.

Also, free attributes enable succinct representation of relative importance of some attributes over others; disabling free attributes thus incurs a loss in succinctness:

Example 7. Consider $n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y$, let $U = \{X_1, X_2, \ldots, X_n\}$, and let $\varphi = \{U \mid y \geq \bar{y}\}$. Then $\varphi^* = \{(uy, u'\bar{y}) \mid u, u' \in \underline{U}\}$, and φ^* is equal to its transitive closure, so, if $o \neq o'$, then $o \succeq_\varphi o'$ if and only if $o[Y] = y$ and $o'[Y] = \bar{y}$. This can be represented, without free attribute, with formula ψ that contains, for every $V \subseteq U$ and every $v \in \underline{V}$, the statement $vy \geq \bar{v}\bar{y}$, where $\bar{v}$ denotes the tuple obtained by inverting all values of v. For every $0 \leq i \leq n$ there are $\binom{n}{i}$ subsets of V of size i, with 2^i ways to choose $v \in \underline{V}$, thus ψ contains $\sum_0^n \binom{n}{i} 2^i = 3^n$ statements.

Restricting to CP-nets induces a further loss in succinctness, as the next example shows:

Example 8. Consider $n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y$, and let φ be the 1-CP$\not\wedge$ formula that contains the following statements: $x_i \geq \bar{x}_i$ for $i = 1, \ldots, n$; $x_1x_2 \ldots x_n : y \geq \bar{y}$; $\bar{x}_i : \bar{y} \geq y$ for $i = 1, \ldots, n$. The size of φ is linear in n. Because preferences for Y depend on all X_i's, a CP-net equivalent to φ will contain, in the table for Y, 2^n CP statements.

Proposition 9. *The following hold:*

- *$\mathcal{L} \ll \mathcal{L}\wedge$ for every $\mathcal{L}$ such that 1-CP$\not\wedge \subseteq \mathcal{L} \subseteq$ CP;*
- *$\mathcal{L} \ll \mathcal{L}\not\wedge$ for every $\mathcal{L}$ such that 1-CP$\wedge \subseteq \mathcal{L} \subseteq$ CP;*
- *1-CP$\not\wedge\wedge \ll$ CPnet.*

6 QUERIES

Table 1 gives an overview of the tractability of the queries that we study in this section. We begin this section with the two queries that have generated most interest in the literature on CP statements.

Linearisability. Knowing that a given $\varphi \in$ CP is linearisable (that is, that $\succeq_\varphi$ is antisymmetric) is valuable, as it makes several other queries easier. It also gives some interesting insights into the semantics of φ. The following query has been addressed in many works on CP statements:[11]

LINEARISABILITY Given φ, is φ linearisable?

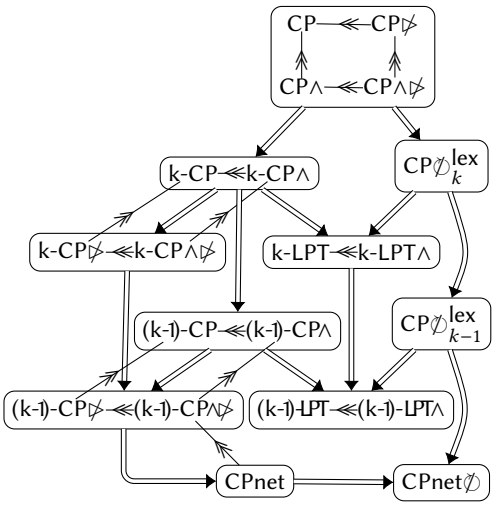
[4] prove that when its dependency graph D_φ is acyclic, then a CP-net φ is linearisable. This result has been extended by [8, 15, 35], who give weaker, sufficient syntactical conditions that guarantee that a locally consistent set of unary, conjunctive CP statements is linearisable; more generally, by definition of k-lexico-compatibility, every formula of CP$\emptyset_k^{\text{lex}}$ is linearisable (since it is compatible with a complete LP-tree). [24, Theorem 3 and 4] prove that LINEARISABILITY is PSPACE-complete for 1-CP$\not\wedge$.

Proposition 10. *LINEARISABILITY can be checked in polynomial time for LPT.*

Comparing alternatives. A basic question, given a formula φ and two alternatives o, o' is: how do o and o' compare, according to φ? Is it the case that $o \succ_\varphi o'$, or $o' \succ_\varphi o$, or $o \bowtie_\varphi o'$, or $o \sim_\varphi o'$? We define the following query, for any relation $R \in \{\succ, \succeq, \sim, \bowtie\}$:

R-COMPARISON Given formula φ, alternatives $o \neq o'$, is it the case that $oR_\varphi o'$?

[11] This query is often called *consistency*.



$\mathcal{L} \Longrightarrow \mathcal{L}''$: $\mathcal{L}$ is strictly more expressive than $\mathcal{L}'$
$\mathcal{L} \ll \mathcal{L}''$: $\mathcal{L}$ is strictly more succinct than $\mathcal{L}'$
For $k > 2$. Boxes contain languages that are equally expressive.

Figure 2: Rel. expressiveness and succinctness

	CP	1-CP▷	1-CP▷∧	CPnet	CP∅ lex k	CPnet∅	CPnet∅ poly	LPT	LTP w. can. tables
LINEARISABILITY	✘✘	✘✘	✘✘		⊤	⊤	⊤	✓	✓
R-COMPARISON, $R \in \{\geq, >, \bowtie\}$	✘✘	✘✘	✘✘	✘○	✘○	✘	✓	✓	✓
∼-COMPARISON	✘✘	✘✘	✘✘		⊥	⊥	⊥	✓	✓
EQUIVALENCE	✘✘	✘○	✘○	✓	✘○	✓	✓	✘○	✓
TOP-p					✓	✓	✓	✓	✓
W. UNDOMINATED ∃	⊤	⊤	⊤	⊤	⊤	⊤	⊤	⊤	⊤
UNDOMINATED ∃	✘✘	✘✘	✘✘		⊤	⊤	⊤	⊤	⊤
S. DOM. ∃, DOM. ∃	✘✘	✘✘				⊤	⊤	✓	✓
UNDOM. CHECK, ≥-CUT EXTRACT.	✓	✓	✓	✓	✓	✓	✓	✓	✓
S. DOM., DOM., W. UNDOM. CHECK	✘✘	✘✘	✘✘			✓	✓	✓	✓
≻-CUT EXTRACTION	✘✘	✘✘	✘✘		✓	✓	✓	✓	✓
≻-CUT COUNTING	✘✘	✘✘	✘✘					✓	✓

Each column corresponds to one sub-language of CP. They are sorted in order of decreasing expressiveness from left to right, except when columns are separated by double lines. For each query and sub-language: ⊤ = always true for the language ; ⊥ = always false for the language; ✓ = polytime answer; ✘ = NP-complete query; ✘○ = NP/coNP-hard query; ✘✘ = PSPACE-complete query.

Table 1: Complexity of queries.

For LP-trees, in order to compare alternatives o and o', one only has to traverse the tree from the root downwards until a node that decides the pair is reached, or up to a leaf if no such node is encountered: in this case o and o' are incomparable. Note that checking if a node decides the pair, and checking if a rule at that nodes applies to order them, can both be done in polynomial time.

PROPOSITION 11. *R-COMPARISON is in P for LPT for $R \in \{>, \geq, \sim, \bowtie\}$.*

Tractability of comparisons, except in some trivial cases, comes at a heavy price in terms of expressiveness: ≥-COMPARISON is tractable for CP-nets when the dependency graph is a polytree [4, Theorem 14], but [4, Theorems 15, 16] prove that ≥-COMPARISON is already NP-hard for the quite restrictive language of binary-valued, directed-path singly connected CP-nets, which are acyclic. [24, Prop. 7, Corrolary 1] prove that ≥-COMPARISON, ≻-COMPARISON, ⋈-COMPARISON and ∼-COMPARISON are PSPACE complete for 1-CP▷∧ and for linearisable, locally complete formulas of 1-CP▷. More precise hardness results for acyclic CP-nets are also proved by [28]. Proposition 12 completes the picture.

PROPOSITION 12. *≻-COMPARISON and ⋈-COMPARISON are NP-hard for the language of fully acyclic CP-nets, and tractable for polytree CP-nets. ∼-COMPARISON is easy for 1-CP $\emptyset^{lex}$.*

Comparing theories. Checking if two theories yield the same preorder can be useful during the compilation process. We say that two formulas φ and φ' are equivalent if they represent the same preorder, that is, if $\succeq_\varphi$ and $\succeq_{\varphi'}$ are identical; we then write $\varphi \equiv \varphi'$.

EQUIVALENCE Given two formulas φ and φ', are they equivalent?

Consider a formula $\varphi \in$ CP, two alternatives o, o', and let $\varphi' = \varphi \cup \{o \geq o'\}$: clearly $o \succeq_{\varphi'} o'$, thus $\varphi \equiv \varphi'$ if and only if $o \succeq_\varphi o'$. Therefore, if language $\mathcal{L}$ is such that adding CP statement $o \geq o'$ to any of its formulas yields a formula that is still in $\mathcal{L}$, then EQUIVALENCE has to be at least as hard as ≥-COMPARISON for $\mathcal{L}$. This is the case of CP. The problem remains hard for 1-CP▷, because it is hard to check the equivalence, in propositional logic, of the conditions of statements that entail a particular swap $x \geq x'$.

Example 9. Consider three attributes A, B and C with respective domains $\{a, \bar{a}\}$, $\{b, \bar{b}\}$ and $\{c_1, c_2, c_3\}$. Consider two CP statements $s = \bar{a} : c_1 \geq c_2$ and $s' = b : c_2 \geq c_3$, and let $\varphi = \{s, s', a : c_1 \geq c_3\}$. Because of statements s and s' we have $\bar{a}bc_1 \geq_\varphi \bar{a}bc_2 \geq_\varphi \bar{a}bc_3$; also, $abc_1 \geq_\varphi abc_3$ because of statement $a : c_1 \geq c_3$. Hence, for any $u \in \underline{A} \times \underline{B}$, if $u \models a \vee (\bar{a}b)$ then $uc_1 \geq uc_3$. Thus $\varphi \equiv \{s, s'\} \cup \{a \vee (\bar{a}b) : c_1 \geq c_3\} \equiv \varphi \cup \{b : c_1 \geq c_3\}$.

PROPOSITION 13. *EQUIVALENCE is coNP-hard for 1-CP▷∧∅, and for 1-LPT∧, both restricted to binary attributes.*

As usual, comparing two formulas is easier for languages where there exists a canonical form. This is the case of CP-nets, as shown by [26, Lemma 2]; their proof makes it clear that the canonical form of any CP-net φ can be computed in polynomial time. Hence:

PROPOSITION 14. *EQUIVALENCE is in P for CP-net.*

EQUIVALENCE also becomes tractable if some form of canonicity is imposed for the conditions of the rules in an LP-tree; this is because, as with CP-net, it is possible to define a canonical form for the structure, by imposing that the labels of the node be as small as possible – which may lead, in some cases, to splitting some nodes.

Top p alternatives. Given a set of alternatives S and some integer p, we may be interested in finding a subset S' of S that contains p "best" alternatives of S, in the sense that for every $o \in S'$, for every

$o' \in S \setminus S'$ it is not the case that $o' \succ_\varphi o$. Note that such a set must exist, because $\succ_\varphi$ is acyclic. The Top-p query is usually defined for totally ordered sets; a definition suited to partial orders is given in [35] (where it is called *ordering*), we adopt this definition here:

Top-p Given $S \subseteq \underline{\mathcal{X}}$, $p < |S|$, and φ, find $o_1, o_2, \ldots, o_p \in S$ such that for every $i \in 1, \ldots, p$, for every $o' \in S$, if $o' \succ_\varphi o_i$ then $o' \in \{o_1, \ldots, o_{i-1}\}$.

Note that if $o_1, o_2, \ldots, o_p$ is the answer to such query, if $1 \leq i < j \leq p$, then it can be the case that $o_i \bowtie o_j$, but it is guaranteed that $o_j \not\succ o_i$: in the context of a recommender system for instance, where one would expect alternatives to be presented in order of non-increasing preference, o_i could be safely presented before o_j.

[4] prove that top-p is tractable for acyclic CP-nets for the specific case where $|S| = 2$. More generally, $\succ$-comparison queries can be used to compute an answer to a top-p query (by asking $\succ$-comparison queries for every pair of elements of S, the number of such pairs being in $\Theta(|S|^2)$). However, [35] shows that an upper approximation of $\succ$ is sufficient, and proves that such an approximation can be obtained in time polynomial in $|\varphi|$ for a restricted class of lexico-compatible formulas of 1-CP$\wedge$ [35, Th. 5]. We prove that this result does indeed hold for the full class of lexico-compatible formulas of 1-CP$\wedge$. The top-p query is also tractable for LPT.

Proposition 15. *top-p can be answered in time which is polynomial in the size of φ and the size of S for k-lexico-compatible formulas (for fixed k); and for LPT.*

Optimization. Instead of ordering a given set, we may want to find a globally optimal alternative. Following [24], given φ, we say that alternative o is:

- weakly undominated if there is no $o' \in \underline{\mathcal{X}}$ such that $o' \succ_\varphi o$;
- undominated if there is no $o' \in \underline{\mathcal{X}}$, $o' \neq o$, such that $o' \succeq_\varphi o$;
- dominating if for every $o' \in \underline{\mathcal{X}}$, $o \succeq_\varphi o'$;
- strongly dominating if for every $o' \in \underline{\mathcal{X}}$ with $o' \neq o$, $o \succ_\varphi o'$.

Note that o is strongly dominating if and only if it is dominating and undominated; and that if o is dominating or undominated, then it is weakly undominated. This gives rise to several types of queries:

[w | s] (undominated | dominating) existence Given φ, is there a [weakly | strongly] (undominated | dominating) alternative?

[w | s] (undominated | dominating) checking Given φ, o, is o a [weakly | strongly] (undominated | dominating) alternative?

All these queries are easily shown to be tractable for LPT. The problem undominated check has been shown to be tractable for CP-nets [4] and for 1-CP$\not\triangleright$ [24]. This can be generalized:

Proposition 16. *undominated check is in P for CP.*

The existence of a weakly undominated alternative is trivially true for CP (in any finite directed graph, at least one vertex has no "strict" predecessor). Linearisability also ensure that there is at least one undominated alternative.

For CP-nets, [4] give a polytime algorithm that computes the only dominating alternative when the dependency graph is acyclic; in this case, this alternative is also the only strongly dominating one and the only undominated one, since the CP-net is linearisable: this implies that dominating $\exists$, s. dominating $\exists$, undominated $\exists$, s. dominating check, dominating check and w. undominated check are tractable for acyclic CP-nets.

[24, Prop. 8, 9 and 11] prove that w. undominated check, dominating check, s. dominating check, dominating $\exists$ and s. dominating $\exists$ are PSPACE-complete for 1-CP$\not\triangleright$, and their reductions for proving hardness of w. undominated check, dominating check, s. dominating check indeed yield formulas of 1-CP$\not\triangleright\wedge$. NP-hardness of undominated $\exists$ for 1-CP$\not\triangleright\wedge$ is proved by [16],

Cuts. *Cuts* are sets of alternatives that are at the same "level" with respect to $\succeq$. For rankings defined with real-valued functions, cuts are defined with respect to possible real values. In the case of preorders, we define cuts with respect to some alternative o: given $\varphi \in$ CP, for any $R \in \{\succ, \succeq\}$, for every alternative o, we define

- $\mathrm{CUT}^{R,o}(\varphi) = \{o' \in \underline{\mathcal{X}} \mid o' \neq o, o' R_\varphi o\}$.

Following [19], we define two families of queries:

R-cut counting Given φ, o, count the elements of $\mathrm{CUT}^{R,o}(\varphi)$

R-cut extraction Given φ, o, return an element of $\mathrm{CUT}^{R,o}(\varphi)$ (or that it is empty)

Proposition 17. *$\succeq$-cut extraction is tractable for CP. $\succ$-cut counting and $\succ$-cut extraction are PSPACE-hard for 1-CP$\not\triangleright\wedge$. For CP$\not\triangleright_k^{\mathsf{lex}}$, $\succ$-cut extraction is equivalent to $\succeq$-cut extraction and is tractable. $\succ$-cut counting is tractable for LP-trees.*

7 CONCLUSION

The literature on languages on CP statements has long focused on statements with unary swaps. Several examples in Section 4 show that this strongly degrades expressiveness. We have introduced a new parameterised family of languages, CP$\not\triangleright_k^{\mathsf{lex}}$, which permits to balance expressiveness against query complexity: the lower k is, the less expressive the language is, but the faster answering most queries will be. Table 1 shows that comparison queries seem to resist tractability, even for CP$\not\triangleright_k^{\mathsf{lex}}$, but queries like the top-p query may be sufficient in many applications. Tractability of the equivalence query relies on the existence of canonical form: it is the case when the language enforces a structure like a dependency graph or a tree, and when the conditions of the statements are restricted to some propositional language with a canonical form.

We have not studied here transformations, like conditioning or other forms of projection for instance. Some initial results on projections can be found in [1]. This is an important direction for future work, as well as properties of the various languages studied here with respect to machine learning.

ACKNOWLEDGMENTS

We thank anonymous referees for their valuable comments on previous versions of this paper. This work has benefited from the AI Interdisciplinary Institute ANITI. ANITI is funded by the French "Investing for the Future – PIA3" program under grant agreement ANR-19-PI3A-0004. This work has also been supported by the PING/ACK project of the French National Agency for Research, grant agreement ANR-18-CE40-0011.

REFERENCES

[1] Philippe Besnard, Jérôme Lang, and Pierre Marquis. 2005. Variable forgetting in preference relations over combinatorial domains. In *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling (MPREF'05).*

[3] Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera. 2010. Learning conditionally lexicographic preference relations. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010) (Frontiers in Artificial Intelligence and Applications, Vol. 215)*, Helder Coelho, Rudi Studer, and Michael Wooldridge (Eds.). IOS Press, 269–274.

[4] Craig Boutilier, Romen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. 2004. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research* 21 (2004), 135–191.

[5] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. 2004. Preference-Based Constrained Optimization with CP-Nets. *Computational Intelligence* 20, 2 (2004), 137–157.

[6] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. 1999. Reasoning With Conditional Ceteris Paribus Preference Statements. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, Kathryn B. Laskey and Henri Prade (Eds.). Morgan Kaufmann, 71–80. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=155&proceeding_id=15

[7] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang. 2009. Conditional Importance Networks: A Graphical Language for Representing Ordinal, Monotonic Preferences over Sets of Goods. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, Craig Boutilier (Ed.). 67–72. http://ijcai.org/Proceedings/09/Papers/022.pdf

[8] Ronen I. Brafman, Carmel Domshlak, and Solomon E. Shimony. 2006. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research* 25 (2006), 389–424.

[9] Darius Braziunas and Craig Boutilier. 2005. Local Utility Elicitation in GAI Models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'05)*, Fahiem Bacchus and Tommi Jaakkola (Eds.). AUAI Press, 42–49.

[10] Michael Bräuning and Eyke Hüllermeyer. 2012. Learning Conditional Lexicographic Preference Trees. In *Preference Learning: Problems and Applications in AI. Proceedings of the ECAI 2012 workshop*, Johannes Fürnkranz and Eyke Hüllermeyer (Eds.). 11–15.

[11] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. 2000. Space Efficiency of Propositional Knowledge Representation Formalisms. *Journal of Artificial Intelligence Research* 13 (2000), 1–31. https://doi.org/10.1613/jair.664

[12] Sylvie Coste-Marquis, Jérôme Lang, Paolo Liberatore, and Pierre Marquis. 2004. Expressive Power and Succinctness of Propositional Languages for Preference Representation, See [17], 203–212. http://www.aaai.org/Library/KR/2004/kr04-023.php

[13] Adnan Darwiche. 1999. Compiling Knowledge into Decomposable Negation Normal Form. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 99)*, Thomas Dean (Ed.). Morgan Kaufmann, 284–289. http://ijcai.org/Proceedings/99-1/Papers/042.pdf

[14] Adnan Darwiche and Pierre Marquis. 2002. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research* 17 (2002), 229–264. https://doi.org/10.1613/jair.989

[15] Carmel Domshlak and Ronen I. Brafman. 2002. CP-nets: Reasoning and Consistency Testing. In *Proceedings of the Eights International Conference on Principles of Knowledge Representation and Reasoning (KR-02)*, Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams (Eds.). Morgan Kaufmann, 121–132.

[16] Carmel Domshlak, Steven David Prestwich, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. 2006. Hard and soft constraints for reasoning about qualitative conditional preferences. *J. Heuristics* 12, 4-5 (2006), 263–285.

[17] Didier Dubois, Christopher A. Welty, and Mary-Anne Williams (Eds.). 2004. *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning*. AAAI Press.

[18] Hélène Fargier, Pierre Francois Gimenez, and Jérôme Mengin. 2018. Learning Lexicographic Preference Trees From Positive Examples. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 2959–2966. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17272/16610

[19] Hélène Fargier, Pierre Marquis, Alexandre Niveau, and Nicolas Schmidt. 2014. A Knowledge Compilation Map for Ordered Real-Valued Decision Diagrams. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, Carla E. Brodley and Peter Stone (Eds.). AAAI Press, 1049–1055. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8195

[20] Hélène Fargier and Jérôme Mengin. 2021. *A Knowledge Compilation Map for Conditional Preference Statements-based Languages*. Research Report IRIT/RR–2021–02–FR. IRIT - Institut de recherche en informatique de Toulouse. https://hal.archives-ouvertes.fr/hal-03133187

[21] Peter C. Fishburn. 1974. Lexicographic Orders, Utilities and Decision Rules: A Survey. *Management Science* 20, 11 (1974), pp. 1442–1471. http://www.jstor.org/stable/2629975

[22] Gerd Gigerenzer and Daniel G. Goldstein. 1996. Reasoning the Fast and Frugal Way: Models of Bounded Rationality. *Psychological Review* 103, 4 (1996), 650–669.

[23] Goran Gogic, Henry A. Kautz, Christos H. Papadimitriou, and Bart Selman. 1995. The Comparative Linguistics of Knowledge Representation, See [29], 862–869. http://ijcai.org/Proceedings/95-1/Papers/111.pdf

[24] Judy Goldsmith, Jérôme Lang, Miroslaw Truszczynski, and Nic Wilson. 2008. The Computational Complexity of Dominance and Consistency in CP-nets. *Journal of Artificial Intelligence Research* 33 (2008), 403–432.

[25] Christophe Gonzales and Patrice Perny. 2004. GAI Networks for Utility Elicitation, See [17], 224–233.

[26] Frédéric Koriche and Bruno Zanuttini. 2010. Learning conditional preference networks. *Artificial Intelligence* 174, 11 (2010), 685–703. https://doi.org/10.1016/j.artint.2010.04.019

[27] Jérome Lang, Jérome Mengin, and Lirong Xia. 2018. Voting on Multi-Issue Domains with Conditionally Lexicographic Preferences. *Artificial Intelligence* 265 (2018), 18–44. https://doi.org/10.1016/j.artint.2018.05.004

[28] Thomas Lukasiewicz and Enrico Malizia. 2019. Complexity results for preference aggregation over (m)CP-nets: Pareto and majority voting. *Artificial Intelligence* 272 (2019), 101–142. https://doi.org/10.1016/j.artint.2018.12.010

[29] Chris S. Mellish (Ed.). 1995. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*. Morgan Kaufmann. http://ijcai.org/proceedings/1995-1

[30] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. 1995. Valued Constraint Satisfaction Problems: Hard and Easy Problems, See [29], 631–639. http://ijcai.org/Proceedings/95-1/Papers/083.pdf

[31] Michael Schmitt and Laura Martignon. 2006. On the Complexity of Learning Lexicographic Strategies. *Journal of Machine Learning Research* 7 (2006), 55–83.

[32] Nic Wilson. 2004. Consistency and Constrained Optimisation for Conditional Preferences. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004)*, Ramón López de Mántaras and Lorenza Saitta (Eds.). IOS Press, 888–892.

[33] Nic Wilson. 2004. Extending CP-Nets with Stronger Conditional Preference Statements. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, Deborah L. McGuinness and George Ferguson (Eds.). AAAI Press / The MIT Press, 735–741.

[34] Nic Wilson. 2006. An Effcient Upper Approximation for Conditional Preference. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006) (Frontiers in Artificial Intelligence and Applications)*, Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso (Eds.). IOS Press.

[35] Nic Wilson. 2011. Computational techniques for a simple theory of conditional preferences. *Artificial Intelligence* 175 (2011), 1053–1091. https://doi.org/10.1016/j.artint.2010.11.018